UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/790,663	03/01/2004	Izydor Gryko	MSFT-2767/305783.01	9551

41505 7590 11/27/2007

WOODCOCK WASHBURN LLP (MICROSOFT CORPORATION)
CIRA CENTRE, 12TH FLOOR
2929 ARCH STREET
PHILADELPHIA, PA 19104-2891

EXAMINER	
WANG, BEN C	

ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
11/27/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/790,663	GRYKO ET AL.	
	Examiner	Art Unit	
	Ben C. Wang	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *05 September 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-5,7-12,14-19 and 21* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-5,7-12,14-19 and 21* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Applicant's amendment dated September 5, 2007, responding to the Office action mailed June 5, 2007 provided in the rejection of claims 1-21, wherein claims 1, 7-8, 14-19, and 21 are amended, claims 6, 13, and 20 are canceled.

Claims 1-5, 7-12, 14-19, and 21 remain pending in the application and which have been fully considered by the examiner.

Applicant's arguments with respect to claims rejection have been fully considered but are moot in view of the new grounds of rejection.

Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Claim Rejections – 35 USC § 103(a)

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

2. Claims 1-5, 7-12, 14-19, and 21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Leach et al. (Pat. No. US 6,412,020 B1) (hereinafter 'Leach') in view of Williams et al. (Pat. No. US 6,256,780 B1) (hereinafter 'Williams')

3. **As to claim 1** (Currently Amended), Leach discloses a method for building an extensible project system (e.g., Abstract, Lines 1-4 – the method aggregates an enclosed object within an enclosing object) comprising:

- providing a base project object comprising data for creating a project system (e.g., Col. 9, Lines 9-11 – enclosing an object within another object while exposing an interface of the enclosed object to client of the enclosing object; Col. 9, Lines 27-30 – to provide a method and system for enclosing objects wherein an enclosed object can itself be an enclosing object to an arbitrary level of enclosing; Col.9, Lines 45-46 – implementing controlling behavior over common functionality present in enclosed objects; Col. 10, Lines 9-13 – an enclosed object is implement with knowledge of the external interfaces of the enclosed object and has no knowledge of interfaces (other than the controlling object management interface; Col. 10, Lines

35-38 – during creation, a pointer to the enclosing multi-type object is passed to the object to be enclosed to enable the enclosed object to communicate with the enclosing multi-type object) of the enclosing object or other enclosed objects);

- providing at least one flavor object comprising data for modifying said project system for a specific purpose (e.g., Col. 9, Lines 9-11 – enclosing an object within another object while exposing an interface of the enclosed object to a client of the enclosing object; Col. 9, Lines 13-14 – enclosing an object within another object after the enclosing object is instantiated; Col. 9, Lines 39-42 – supplying default functionality to objects by enclosing them within an enclosing object where an enclosed or enclosing object implements the default functionality; Col. 9, Lines 55-58 – the enclosed object has an object management interface and on or more external interfaces, while the enclosing object has a controlling object management interface); and
- creating a flavored project system adapted for said specific purpose by object aggregation using said base project object as a participating object and one of said at least one flavor objects as a controlling object (e.g., Col. 8, Lines 66-67 – a method and system for aggregating objects; Col. 9, Lines 4-7 – dynamically aggregating objects; statically aggregating objects; Col. 9, Lines 25-26 – implementing an aggregate object so that a client is unaware that the object is an aggregate; Col. 9, Lines 50-61 – the method aggregates an enclosed object within an enclosing object; each interface exposed to a client by the aggregate object has a query function member for receiving an identifier of an interface and for returning a

reference to the identified interface; Col. 10, Lines 8-13 (static aggregation), 17-24 (dynamic aggregation); Col. 10, Lines 24-30 – the multi-type object has an add interface function member, which can be used to aggregate interfaces by adding them to the enclosing multi-type object; the multi-type object also has an add object function member for aggregating all of the interface of an object; Col. 10, 34-47 – a preferred method invokes the add interface function member or the add object function member of the enclosing multi-type object passing it a reference to the created object implementing the interface to be aggregated; the query function member of the enclosing multi-type object is invoked in order to retrieve a reference to the interface that has been aggregated).

Leach does not explicitly disclose the followings:

- wherein the base project object implements a base project configuration object that includes configuration properties for the base project object;
- wherein the at least one flavor object includes flavor-specific project configuration properties;
- signaling by the base project object to the at least one flavor object that the base project configuration object needs to be extended; and
- creating, by the at least one flavor object, a flavored base project configuration object, wherein at least one configuration property for the base project object is modified by a corresponding flavor-specific project configuration property.

However, in an analogous art of *method and system for assembling software components*, Williams discloses the followings:

- wherein the base project object implements a base project configuration object that includes configuration properties for the base project object;
- wherein the at least one flavor object includes flavor-specific project configuration properties (e.g., Col. 8, Line 57 through Col. 9, Line 2 - The *IinitFromTemplateStream* interface of an assembly object (i.e., a base project configuration object) has one method that controls the initialization of the assembly object from a passed stream; The initialization data is static configuration data along with the initialization data for assembly parameters. Other assembly data, such as ambient properties, can be passed to an assembly via a connection. The initialization data for the assembly parameters is passed in the steam and the static configuration data can be available through the class identifier of the assembly. The assembly can be customized through the parameters);
- signaling by the base project object to the at least one flavor object that the base project configuration object needs to be extended (e.g., Fig. 1 – illustrating an assembly object along with its connection to external entities; Col. 5, Lines 4-6 – Fig. 1 illustrates the interconnection of *assembly3* (i.e., the base project object) with external entities (i.e., one flavor object), that is , *assebmly1 101* and *assembly2 102*); and

- creating, by the at least one flavor object, a flavored base project configuration object, wherein at least one configuration property for the base project object is modified by a corresponding flavor-specific project configuration property (i.e., Fig. 5 – detailed block diagram of a sample implementation of *assembly3* of Fig. 1; Col. 9, Lines 23-27 - ...and exposes the *IinitFromTemplateStream* interface through which the assembly can be initialized; Col. 9, Line 64 through Col. 10, Line 3 – In step 603, if the first element supports the *IinitFromTemplateStream* interface, then the processes invokes the method Init of that interface passing a reference to the string that contains the initialization data (i.e., configuration property) for the element; Col. 12, Lines 39-45 – The PCODE interpreter is implemented by the method *IinitFromTemplateStream* of the *IinitFromTemplateStream* interface of the assembly object. The PCODE interpreter initializes the assembly object in accordance with the pseudo-code for the assembly object as modified by assembly parameters passed via the stream to the method).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Williams into the Leach's system to further provide the followings in Leach's system:

- wherein the base project object implements a base project configuration object that includes configuration properties for the base project object;
- wherein the at least one flavor object includes flavor-specific project configuration properties;

- signaling by the base project object to the at least one flavor object that the base project configuration object needs to be extended; and
- creating, by the at least one flavor object, a flavored base project configuration object, wherein at least one configuration property for the base project object is modified by a corresponding flavor-specific project configuration property.

The motivation is that it would further enhance the Leach's system by taking, advancing and/or incorporating Williams's system which offers significant advantages for providing a light-weight mechanism for instantiating the component objects of an assembly object and a light-weight mechanism for controlling the exporting and importing of objects; thus, the assembly system relieves the developer of an assembly object from the task of developing code for such instantiation and control as once suggested by Williams (e.g., Col. 2, Lines 37-44).

4. **As to claim 2** (Original) (incorporating the rejection in claim 1), Leach discloses the method where said at least one flavor object comprises at least a first flavor object and a second flavor object, and where said step of creating a flavored project system comprises: creating an intermediary object by aggregating said first flavor object as a controlling object and said base project object as a participating object; and creating a flavored project system by using said second flavor object as a controlling object and said intermediary object as a participating object (e.g., Col. 9, Lines 27-30 – to provide a method and system for enclosing objects where an enclosed object can itself be an enclosing object to an arbitrary level of enclosing).

5. **As to claim 3** (Original) (incorporating the rejection in claim 1), Leach discloses the method where said step of creating a flavored project system comprises allowing at least one interface of said base project to be modified by said flavor object (e.g., Col. 9, Lines 34-37 – for enhancing a base object's apparent behavior by adding an interface to it that overrides standard behavior of the base object; Col. 25, Lines 3-16 – these combining rules can be used to override the standard behavior of an enclosed base object by providing access to a new implementation of a previously defined interface of the enclosed base object).

6. **As to claim 4** (Original) (incorporating the rejection in claim 3), Leach discloses the method where said step of creating a flavored project system comprises allowing a value for at least one property stored in said at least one interface of said base project to be modified by a value for said at least one property stored in an interface of said flavor object (e.g., Col. 5, Lines 10-12 – the overriding virtual function can modify the state of the object in a way that affects non-overridden functions; Col. 9, Lines 1-2 – to provide a method and system for dynamically modifying object behavior; Col. 9, Lines 34-37 – for enhancing a base object's apparent behavior by adding an interface to it that overrides standard behavior of the base object; Col. 25, Lines 3-16 – these combining rules can be used to override the standard behavior of an enclosed base object by providing access to a new implementation of a previously defined interface of the enclosed base object; Col. 10, Lines 17-24 – an object can be modified dynamically by

allowing interface instances, as implemented by objects, to be aggregated during the execution of a client program).

7. **As to claim 5** (Original) (incorporating the rejection in claim 1), Leach discloses the method where said step of creating a flavored project system comprises allowing at least one interface of said base project to be replaced by said flavor object (e.g., Col. 9, Lines 34-37 – for enhancing a base object's apparent behavior by adding an interface to it that overrides standard behavior of the base object; Col. 25, Lines 3-16 – these combining rules can be used to override the standard behavior of an enclosed base object by providing access to a new implementation of a previously defined interface of the enclosed base object).

8. **As to claim 7** (Currently Amended) (incorporating the rejection in claim 1), Williams discloses the method wherein said flavored base project configuration object (e.g., Col. 8, Line 57 through Col. 9, Line 2 - The *IinitFromTemplateStream* interface of an assembly object (i.e., a base project configuration object) has one method that controls the initialization of the assembly object from a passed stream; The initialization data is static configuration data along with the initialization data for assembly parameters. Other assembly data, such as ambient properties, can be passed to an assembly via a connection. The initialization data for the assembly parameters is passed in the steam and the static configuration data can be available through the class identifier of the assembly. The assembly can be customized through the parameters)

includes an extender interface, said creation of a project system further comprising: providing an extender site object associated with said extender interface (e.g., Fig. 1 – illustrating an assembly object along with its connection to external entities; Col. 4, Line 52 through Col. 5, Line 20 – an external entity connects assembly-2 to assembly-3 by retrieving the reference to a connector of assembly-2 and requesting the connector to export the element identified by index "i1", represented by plug102a. the external entity then requests connector-3 of assembly-3 to connect assembly-2 through the connection identified by role "r1", represented by socket 103b).

9. **As to claim 8** (Currently Amended), Leach discloses a system for building an extensible project system (Abstract, Lines 1-4 – the method aggregates an enclosed object within an enclosing object) comprising:

- A process configured to instantiate a base project object comprising data for creating a project system (e.g., Col. 9, Lines 9-11 – enclosing an object within another object while exposing an interface of the enclosed object to client of the enclosing object; Col. 9, Lines 27-30 – to provide a method and system for enclosing objects wherein an enclosed object can itself be an enclosing object to an arbitrary level of enclosing; Col.9, Lines 45-46 – implementing controlling behavior over common functionality present in enclosed objects; Col. 10, Lines 9-13 – an enclosed object is implement with knowledge of the external interfaces of the enclosed object and has no knowledge of interfaces (other than the controlling object management interface; Col. 10, Lines 35-38 – during creation, a pointer to the enclosing multi-type object is

passed to the object to be enclosed to enable the enclosed object to communicate with the enclosing multi-type object) of the enclosing object or other enclosed objects);

- A process configured to instantiate at least one flavor object comprising data for modifying said project system for a specific purpose (e.g., Col. 9, Lines 9-11 – enclosing an object within another object while exposing an interface of the enclosed object to a client of the enclosing object; Col. 9, Lines 13-14 – enclosing an object within another object after the enclosing object is instantiated; Col. 9, Lines 39-42 – supplying default functionality to objects by enclosing them within an enclosing object where an enclosed or enclosing object implements the default functionality; Col. 9, Lines 55-58 – the enclosed object has an object management interface and on or more external interfaces, while the enclosing object has a controlling object management interface); and
- a process configured to generate a flavored project system for said specific purpose by object aggregation using said base project object as a participating object and one of said at least one flavor objects as a controlling object (e.g., Col. 8, Lines 66-67 – a method and system for aggregating objects; Col. 9, Lines 4-7 – dynamically aggregating objects; statically aggregating objects; Col. 9, Lines 25-26 – implementing an aggregate object so that a client is unaware that the object is an aggregate; Col. 9, Lines 50-61 – the method aggregates an enclosed object within an enclosing object; each interface exposed to a client by the aggregate object has a query function member for receiving an identifier of an interface and for returning a

reference to the identified interface; Col. 10, Lines 8-13 (static aggregation), 17-24 (dynamic aggregation); Col. 10, Lines 24-30 – the multi-type object has an add interface function member, which can be used to aggregate interfaces by adding them to the enclosing multi-type object; the multi-type object also has an add object function member for aggregating all of the interface of an object; Col. 10, 34-47 – a preferred method invokes the add interface function member or the add object function member of the enclosing multi-type object passing it a reference to the created object implementing the interface to be aggregated; the query function member of the enclosing multi-type object is invoked in order to retrieve a reference to the interface that has been aggregated).

Leach does not explicitly disclose the followings:

- wherein the base project object implements a base project configuration; object that includes configuration properties for the base project object;
- wherein the at least one flavor object includes flavor-specific project configuration properties;
- a process in the base project object configured to signal the at least one flavor object that the base project configuration object needs to be extended; and
- a process in the at least one flavor object configured to generate a flavored base project configuration object, wherein at least one configuration property for the base project object is modified by a corresponding flavor-specific project configuration property.

However, in an analogous art of *method and system for assembling software components*, Williams discloses the followings:

- wherein the base project object implements a base project configuration; object that includes configuration properties for the base project object;
- wherein the at least one flavor object includes flavor-specific project configuration properties (e.g., Col. 8, Line 57 through Col. 9, Line 2 - The *IinitFromTemplateStream* interface of an assembly object (i.e., a base project configuration object) has one method that controls the initialization of the assembly object from a passed stream; The initialization data is static configuration data along with the initialization data for assembly parameters. Other assembly data, such as ambient properties, can be passed to an assembly via a connection. The initialization data for the assembly parameters is passed in the steam and the static configuration data can be available through the class identifier of the assembly. The assembly can be customized through the parameters);
- a process in the base project object configured to signal the at least one flavor object that the base project configuration object needs to be extended (e.g., Fig. 1 – illustrating an assembly object along with its connection to external entities; Col. 5, Lines 4-6 – Fig. 1 illustrates the interconnection of *assembly3* (i.e., the base project object) with external entities (i.e., one flavor object), that is , *assebmly1 101* and *assembly2 102*); and

- a process in the at least one flavor object configured to generate a flavored base project configuration object, wherein at least one configuration property for the base project object is modified by a corresponding flavor-specific project configuration property (i.e., Fig. 5 – detailed block diagram of a sample implementation of *assembly3* of Fig. 1; Col. 9, Lines 23-27 - ...and exposes the *IinitFromTemplateStream* interface through which the assembly can be initialized; Col. 9, Line 64 through Col. 10, Line 3 – In step 603, if the first element supports the *IinitFromTemplateStream* interface, then the processes invokes the method Init of that interface passing a reference to the string that contains the initialization data (i.e., configuration property) for the element; Col. 12, Lines 39-45 – The PCODE interpreter is implemented by the method *IinitFromTemplateStream* of the *IinitFromTemplateStream* interface of the assembly object. The PCODE interpreter initializes the assembly object in accordance with the pseudo-code for the assembly object as modified by assembly parameters passed via the stream to the method).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Williams into the Leach's system to further provide the followings in Leach's system:

- wherein the base project object implements a base project configuration; object that includes configuration properties for the base project object;
- wherein the at least one flavor object includes flavor-specific project configuration properties;

- a process in the base project object configured to signal the at least one flavor object that the base project configuration object needs to be extended; and
- a process in the at least one flavor object configured to generate a flavored base project configuration object, wherein at least one configuration property for the base project object is modified by a corresponding flavor-specific project configuration property.

The motivation is that it would further enhance the Leach's system by taking, advancing and/or incorporating Williams's system which offers significant advantages for providing a light-weight mechanism for instantiating the component objects of an assembly object and a light-weight mechanism for controlling the exporting and importing of objects; thus, the assembly system relieves the developer of an assembly object from the task of developing code for such instantiation and control as once suggested by Williams (e.g., Col. 2, Lines 37-44).

10. **As to claim 9** (Original) (incorporating the rejection in claim 8), Leach discloses the system where said at least one flavor object comprises at least a first flavor object and a second flavor object, and where said aggregator further comprises: a first aggregator for creating an intermediary object by aggregating said first flavor object as a controlling object and said base project object as a participating object; and a second aggregator for creating a flavored project system by using said second flavor object as a controlling object and said intermediary object as a participating object (e.g., Col. 9,

Lines 27-30 – to provide a method and system for enclosing objects where an enclosed object can itself be an enclosing object to an arbitrary level of enclosing).

11. **As to claim 10** (Original) (incorporating the rejection in claim 8), Leach discloses the system where said aggregator causes at least one interface of said base project to be modified by said flavor object (e.g., Col. 9, Lines 34-37 – for enhancing a base object's apparent behavior by adding an interface to it that overrides standard behavior of the base object; Col. 25, Lines 3-16 – these combining rules can be used to override the standard behavior of an enclosed base object by providing access to a new implementation of a previously defined interface of the enclosed base object).

12. **As to claim 11** (Original) (incorporating the rejection in claim 10), Leach discloses the system where said aggregator causes a value for at least one property stored in said at least one interface of said base project to be modified by a value for said at least one property stored in an interface of said flavor object (e.g., Col. 5, Lines 10-12 – the overriding virtual function can modify the state of the object in a way that affects non-overridden functions; Col. 9, Lines 1-2 – to provide a method and system for dynamically modifying object behavior; Col. 9, Lines 34-37 – for enhancing a base object's apparent behavior by adding an interface to it that overrides standard behavior of the base object; Col. 25, Lines 3-16 – these combining rules can be used to override the standard behavior of an enclosed base object by providing access to a new implementation of a previously defined interface of the enclosed base object; Col. 10,

Lines 17-24 – an object can be modified dynamically by allowing interface instances, as implemented by objects, to be aggregated during the execution of a client program).

13. **As to claim 12** (Original) (incorporating the rejection in claim 8), Leach discloses the system where said aggregator causes at least one interface of said base project to be replaced by said flavor object (e.g., Col. 9, Lines 34-37 – for enhancing a base object's apparent behavior by adding an interface to it that overrides standard behavior of the base object; Col. 25, Lines 3-16 – these combining rules can be used to override the standard behavior of an enclosed base object by providing access to a new implementation of a previously defined interface of the enclosed base object).

14. **As to claim 14** (Currently Amended) (incorporating the rejection in claim 8), Williams discloses the system wherein said flavored base project configuration object (e.g., Col. 8, Line 57 through Col. 9, Line 2 - The *IinitFromTemplateStream* interface of an assembly object (i.e., a base project configuration object) has one method that controls the initialization of the assembly object from a passed stream; The initialization data is static configuration data along with the initialization data for assembly parameters. Other assembly data, such as ambient properties, can be passed to an assembly via a connection. The initialization data for the assembly parameters is passed in the steam and the static configuration data can be available through the class identifier of the assembly. The assembly can be customized through the parameters) includes an extender interface, said project system further comprising: an extender site

object associated with said extender interface (e.g., Fig. 1 – illustrating an assembly object along with its connection to external entities; Col. 4, Line 52 through Col. 5, Line 20 – an external entity connects assembly-2 to assembly-3 by retrieving the reference to a connector of assembly-2 and requesting the connector to export the element identified by index "i1", represented by plug102a. the external entity then requests connector-3 of assembly-3 to connect assembly-2 through the connection identified by role"r1", represented by socket 103b).

15. **As to claim 15** (Currently Amended), Leach discloses a computer-readable medium storage for building an extensible project system (e.g., Abstract, Lines 1-4 – the method aggregates an enclosed object within an enclosing object), said computer readable storage medium storing instructions for causing a computer to perform the steps of comprising:

- providing a base project object comprising data for creating a project system (e.g., Col. 9, Lines 9-11 – enclosing an object within another object while exposing an interface of the enclosed object to client of the enclosing object; Col. 9, Lines 27-30 – to provide a method and system for enclosing objects wherein an enclosed object can itself be an enclosing object to an arbitrary level of enclosing; Col.9, Lines 45-46 – implementing controlling behavior over common functionality present in enclosed objects; Col. 10, Lines 9-13 – an enclosed object is implement with knowledge of the external interfaces of the enclosed object and has no knowledge of interfaces (other than the controlling object management interface; Col. 10, Lines

35-38 – during creation, a pointer to the enclosing multi-type object is passed to the object to be enclosed to enable the enclosed object to communicate with the enclosing multi-type object) of the enclosing object or other enclosed objects);

- providing at least one flavor object comprising data for modifying said project system for a specific purpose (e.g., Col. 9, Lines 9-11 – enclosing an object within another object while exposing an interface of the enclosed object to a client of the enclosing object; Col. 9, Lines 13-14 – enclosing an object within another object after the enclosing object is instantiated; Col. 9, Lines 39-42 – supplying default functionality to objects by enclosing them within an enclosing object where an enclosed or enclosing object implements the default functionality; Col. 9, Lines 55-58 – the enclosed object has an object management interface and on or more external interfaces, while the enclosing object has a controlling object management interface); and
- creating a flavored project system for said specific purpose by object aggregation using said base project object as a participating object and one of said at least one flavor objects as a controlling object (Col. 8, Lines 66-67 – a method and system for aggregating objects; Col. 9, Lines 4-7 – dynamically aggregating objects; statically aggregating objects; Col. 9, Lines 25-26 – implementing an aggregate object so that a client is unaware that the object is an aggregate; Col. 9, Lines 50-61 – the method aggregates an enclosed object within an enclosing object; each interface exposed to a client by the aggregate object has a query function member for receiving an identifier of an interface and for returning a reference to the identified interface; Col.

10, Lines 8-13 (static aggregation), 17-24 (dynamic aggregation); Col. 10, Lines 24-30 – the multi-type object has an add interface function member, which can be used to aggregate interfaces by adding them to the enclosing multi-type object; the multi-type object also has an add object function member for aggregating all of the interface of an object; Col. 10, 34-47 – a preferred method invokes the add interface function member or the add object function member of the enclosing multi-type object passing it a reference to the created object implementing the interface to be aggregated; the query function member of the enclosing multi-type object is invoked in order to retrieve a reference to the interface that has been aggregated).

Leach does not explicitly disclose the followings:

- wherein the base project implements a base project configuration object that includes configuration properties for the base project object;
- wherein the at least one flavor object includes flavor-specific project configuration properties;
- signaling by the base project object to the at least one flavor object that the base project configuration object needs to be extended; and
- creating, by the at least one flavor object, a flavored base project configuration object, wherein at least one configuration property for the base project object is modified by a corresponding flavor-specific project configuration property.

However, in an analogous art of *method and system for assembling software components*, Williams discloses the followings:

- wherein the base project implements a base project configuration object that includes configuration properties for the base project object;
- wherein the at least one flavor object includes flavor-specific project configuration properties (e.g., Col. 8, Line 57 through Col. 9, Line 2 - The *IinitFromTemplateStream* interface of an assembly object (i.e., a base project configuration object) has one method that controls the initialization of the assembly object from a passed stream; The initialization data is static configuration data along with the initialization data for assembly parameters. Other assembly data, such as ambient properties, can be passed to an assembly via a connection. The initialization data for the assembly parameters is passed in the steam and the static configuration data can be available through the class identifier of the assembly. The assembly can be customized through the parameters);
- signaling by the base project object to the at least one flavor object that the base project configuration object needs to be extended (e.g., Fig. 1 – illustrating an assembly object along with its connection to external entities; Col. 5, Lines 4-6 – Fig. 1 illustrates the interconnection of *assembly3* (i.e., the base project object) with external entities (i.e., one flavor object), that is , *assebmly1 101* and *assembly2 102*); and
- creating, by the at least one flavor object, a flavored base project configuration object, wherein at least one configuration property for the base project object is modified by a corresponding flavor-specific project

configuration property (i.e., Fig. 5 – detailed block diagram of a sample implementation of *assembly3* of Fig. 1; Col. 9, Lines 23-27 - ...and exposes the *IinitFromTemplateStream* interface through which the assembly can be initialized; Col. 9, Line 64 through Col. 10, Line 3 – In step 603, if the first element supports the *IinitFromTemplateStream* interface, then the processes invokes the method Init of that interface passing a reference to the string that contains the initialization data (i.e., configuration property) for the element; Col. 12, Lines 39-45 – The PCODE interpreter is implemented by the method *IinitFromTemplateStream* of the *IinitFromTemplateStream* interface of the assembly object. The PCODE interpreter initializes the assembly object in accordance with the pseudo-code for the assembly object as modified by assembly parameters passed via the stream to the method).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Williams into the Leach's system to further provide the followings in Leach's system:

- wherein the base project implements a base project configuration object that includes configuration properties for the base project object;
- wherein the at least one flavor object includes flavor-specific project configuration properties;
- signaling by the base project object to the at least one flavor object that the base project configuration object needs to be extended; and

- creating, by the at least one flavor object, a flavored base project configuration object, wherein at least one configuration property for the base project object is modified by a corresponding flavor-specific project configuration property.

The motivation is that it would further enhance the Leach's system by taking, advancing and/or incorporating Williams's system which offers significant advantages for providing a light-weight mechanism for instantiating the component objects of an assembly object and a light-weight mechanism for controlling the exporting and importing of objects; thus, the assembly system relieves the developer of an assembly object from the task of developing code for such instantiation and control as once suggested by Williams (e.g., Col. 2, Lines 37-44).

16. **As to claim 16** (Currently Amended) (incorporating the rejection in claim 15), Leach discloses the computer-readable medium storage where said at least one flavor object comprises at least a first flavor object and a second flavor object, and where said step of creating a flavored project system comprises: creating an intermediary object by aggregating said first flavor object as a controlling object and said base project object as a participating object; and creating a flavored project system by using said second flavor object as a controlling object and said intermediary object as a participating object (e.g., Col. 9, Lines 27-30 – to provide a method and system for enclosing objects where an enclosed object can itself be an enclosing object to an arbitrary level of enclosing).

17. **As to claim 17** (Currently Amended) (incorporating the rejection in claim 15), Leach discloses the computer-readable medium storage where said step of creating a flavored project system comprises allowing at least one interface of said base project to be modified by said flavor object (e.g., Col. 9, Lines 34-37 – for enhancing a base object's apparent behavior by adding an interface to it that overrides standard behavior of the base object; Col. 25, Lines 3-16 – these combining rules can be used to override the standard behavior of an enclosed base object by providing access to a new implementation of a previously defined interface of the enclosed base object).

18. **As to claim 18** (Currently Amended) (incorporating the rejection in claim 17), Leach discloses the computer-readable medium storage where said step of creating a flavored project system comprises allowing a value for at least one property stored in said at least one interface of said base project to be modified by a value for said at least one property stored in an interface of said flavor object (e.g., Col. 5, Lines 10-12 – the overriding virtual function can modify the state of the object in a way that affects non-overridden functions; Col. 9, Lines 1-2 – to provide a method and system for dynamically modifying object behavior; Col. 9, Lines 34-37 – for enhancing a base object's apparent behavior by adding an interface to it that overrides standard behavior of the base object; Col. 25, Lines 3-16 – these combining rules can be used to override the standard behavior of an enclosed base object by providing access to a new implementation of a previously defined interface of the enclosed base object; Col. 10,

Lines 17-24 – an object can be modified dynamically by allowing interface instances, as implemented by objects, to be aggregated during the execution of a client program).

19. **As to claim 19** (incorporating the rejection in claim 15), Leach discloses the computer-readable medium storage where said step of creating a flavored project system comprises allowing at least one interface of said base project to be replaced by said flavor object (e.g., Col. 9, Lines 34-37 – for enhancing a base object's apparent behavior by adding an interface to it that overrides standard behavior of the base object; Col. 25, Lines 3-16 – these combining rules can be used to override the standard behavior of an enclosed base object by providing access to a new implementation of a previously defined interface of the enclosed base object).

20. **As to claim 21** (Currently Amended) (incorporating the rejection in claim 15), Williams discloses the computer-readable storage medium wherein said flavored base project configuration object (e.g., Col. 8, Line 57 through Col. 9, Line 2 - The *IinitFromTemplateStream* interface of an assembly object (i.e., a base project configuration object) has one method that controls the initialization of the assembly object from a passed stream; The initialization data is static configuration data along with the initialization data for assembly parameters. Other assembly data, such as ambient properties, can be passed to an assembly via a connection. The initialization data for the assembly parameters is passed in the steam and the static configuration data can be available through the class identifier of the assembly. The assembly can be customized through the parameters) includes an extender interface, said creation of a

project system further comprising: providing an extender site object associated with said extender interface (e.g., Fig. 1 – illustrating an assembly object along with its connection to external entities; Col. 4, Line 52 through Col. 5, Line 20 – an external entity connects assembly-2 to assembly-3 by retrieving the reference to a connector of assembly-2 and requesting the connector to export the element identified by index "i1", represented by plug102a. the external entity then requests connector-3 of assembly-3 to connect assembly-2 through the connection identified by role "r1", represented by socket 103b).

Conclusion

21. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is 571-270-1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

TUAN DAM
SUPERVISORY PATENT EXAMINER

BCW

November 20, 2007